Entered ✓

WINDOWS BASED BLOCK DIAGRAM SIMULATION FOR DSP SYSTEMS : II-IMPLEMENTATION OF SIMULATION BLOCKS

by
MAJOR SURESH WARRIER

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

MARCH, 1995

WINDOWS BASED BLOCK DIAGRAM SIMULATION FOR DSP SYSTEMS : II -- IMPLEMENTATION OF SIMULATION BLOCKS

A thesis submitted
in partial fulfilment of the requirements
for the degree of

Master of Technology

by

Major Suresh Warrier

to the

Department of Electrical Engineering
Indian Institute of Technology , Kanpur
March 1995

EE-1995-M-WAR-WIN

CERTIFICATE

It is certified that the work outlined in this thesis titled **Windows Based Block Diagram Simulation for DSP Systems :II- Implementation of Simulation Blocks** by Major Suresh Warrier has been carried out under my supervision and that this work has not been submitted elsewhere for a degree .

Anil Mahanta

Dr. Anil Mahanta

Dept. of EE

IIT Kanpur

March 1995

ABSTRACT

In this thesis a library of DSP and mathematical functions have been developed and implementation of simulation of blocks to form part of a Windows based block diagram simulation package has been done . The graphical user interface for the same has been implemented concurrently in another thesis. The software is capable of simulation of a variety of DSP functions and the output can be viewed on a graphical display.The software developed is user friendly with on - line help. The software is intended to be used as a teaching aid for conceptualization and experimentation of DSP systems.

Acknowledgements

I would like to express my sincere gratitude to Dr Anil Mahanta for the help and guidance he provided. I would remain indebted to him for the patience he displayed during the trying moments and the pains he undertook during the preparation of this report.

I would also like to specially thank the defence officers' community for making the stay at the IIT a memorable one. I also want to thank the M Tech (EE) batch of 93 for their friendship and valuable advice and help at times of need.

Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 General

Simulation has come to play a major role in the design , analysis and implementation of signal processing systems . The vast improvement in performanceof computers has resulted in the development of sophisticated software for mathematical problem solving at desktop level. The software has been improving over the years and vendors have been adding more functionalities to make their product more competitive . A welcome addition in these functionalities has been that of the signal processing Toolboxes. Some of the popular packages which have entered into the domain of signal processing are Matlab with its signal processing Toolbox , Gabriel and Hypersignal .

Another major development has been the improvement in the display of data and results with more powerful and user interactive graphic tools for display . These tools have improved the aspect of visualization of the data into newer ways which are now available on desktops . Visual data analysis (VDA) [1] is now moving from a select preserve of reseachers and scientists to the level of engineers and designers . The widespread use of Windows all over the world is a proof of the popularity of such GUI (graphical user interface). The wide acceptance of Windows platform is evident from the release of Windows versions of almost all popular commercial and scientific software. In fact , this has also been the reason de 'etre for adoption of the Windows environment for this thesis in keeping with these current trends .

A feature of the various simulation software packages has been the merger of the number-crunching and symbolic abilities with better front-end presentation and better manipulation of graphics , equations and text . The simulation of DSP systems based on block diagram has been the approach in Hypersignal [1] Gabriel [2] is another system in which block diagram simulation of functional DSP blocks is possible . Matlab with its signal processing Toolbox [3]can also simulate the DSP environment . There are many other software packages which perform similar tasks of design , implementation and simulation of block diagrams of DSP systems . Information regarding these have been given in [1].

1.2 Objectives and Scope of the Thesis

The objective of the thesis was as under :

- To develop a Windows based package for simulation of DSP systems .

  The purpose of this software could be for analysis of DSP systems and as a teaching aid for the concepts of DSP.

Towards this objective , the task has been split into the following modules.

- Development of the visual user interface
- Development of function library to carry out the simulation.
- Design of a graphical module for representation of the simulation results .

These would be followed by integration of the individual modules.

The complete software has been developed in two theses concurrently.The development of the user interface and design of the graphical module has been done in [4].The scope of this thesis has been to develop a library of mathematical and DSP functions and evolving the methodology of execution of the simulation of blocks.

1.3 A Brief Overview

Windows is a new entry in the realm of user friendly graphical user interface Presently it provides a platform above the operating system In fact , it is poised to take over the reins of an operating system with its forthcoming release .

The concepts of object oriented programming (OOP) are fast becoming the pillars of any programming activity. This provides the programmer several advantages like encapsulation and polymorphism . C++ has been developed in order to provide the programmer the facilities offered by C programming in the OOP environment . Borland C++ version 4.0 and its higher versions provide a vast gamut of functions to invoke the power of Windows in order to develop large scale applications using OOP.

The above mentioned ideas have been the major motivation in the choice of this environment for the development of this application . The software code has been written in C++ using the concepts of OOP . The application uses the in built libraries for the development of window structures .

The software is a menu - driven package . The user can select the required blocks for the simulation and place them on the screen.He can then connect them in the order he desires , and then run the simulation . He can then view the output data of any block in the chain by choosing the display option . He is given the flexibility to change the parameters if he wishes to do so , run the simulator any number of times or clean the previous diagram and start a fresh one.

Thus the software provides a viable software experimentation tool for a student in order to study the nature and characteristics of a DSP system . He can easily learn the aspects of design of systems as he can study the responses to various types of signals which can be generated by the system or taken from the user by file.

1.4 Development of Object Windows based GUI

This part of the software has been developed in [4]. However , a brief description of the same is in order as it pertains to this work. The GUI has been developed using the class library OWL 2.0 (Object Windows Library 2.0) which provides a platform to effectively utilize the power of C++ viz, encapsulation, inheritance and polymorphism. All the components of the GUI have been created using the existing classes of OWL 2.0 or the class created after inherting and customization of base classes. For a detailed description of these topics, refer to Chapter 3 of [4].

1.5 Organization of the Thesis

The thesis is organized under the following heads

- Chapter 2 : Working of the Simulator . This chapter covers the data structures used in the application and the method of executing the simulation.
- Chapter 3 : Implementation of DSP and Mathematical Functions . This chapter contains the description of the implementation of DSP and mathematical functions and an example of simulation.
- Chapter 4 : Enhancements to be done in the software and brief outline of approach to be adopted for incorporation of new functions.
- Chapter 5 : Conclusions.
- Appendix A : Users' Guide for DSP and Mathematical Functions.
- Bibliography

CHAPTER 2

WORKING OF THE SIMULATOR

The simulation software has been developed in Borland C++ Version 4.02 environment . The data structure of the simulation software has been created in two layers. Each layer has a unique class to represent its data structure . The first layer, henceforth referred to as the foreground layer consists of structure to cater for operations occurring on the screen. The second layer referred to as the background layer consists of a different data structure for the actual data handling and processing in the DSP and other mathematical functions. The foreground layer and its data structure have been localized in function to the Client window of the GUI. However, the data structure of the background layer is accessible from all the modules .

In the following sections we discuss the details of these layers and their functioning when a simulation run takes place .

2.1 The Foreground Layer.

This layer consists of objects that are directly connected to the screen . This layer manages several tasks depending on the screen based commands given by the user through the main application menu or the speed-bar menu.

- **Creation**

The first task is the creation of the object based on the type of the block chosen from the menu.This creates a child window on the screen. An icon representing the type of block chosen is placed in this window which formally depicts the type of block chosen by the user. The name of the block chosen also apppears at the bottom of the icon .

- **Block selection**

To perform any action using a block it has to be made active on the screen which is conveniently done by clicking the mouse on the concerned block. This is necessary in order to address that block for connection, changing parameters or display.

- **Connection**

Whenever the user draws a connection between the blocks linking is done through a linked list.The program automatically checks for any wrong connection and accordingly warns the user.The check for detecting a feedback loop is done by this layer as the occurrence of feedback would change the way the application implements the simulation run.

- **Changing Parameters**

In case the user wishes to change the parameters of any particular block then the background data structure of the selected child window chosen is called. This action is done in the foreground layer.

2.2 The Background Layer.

This layer lies in the background and consists of a class BLOCK . This generates objects which handle the actual input and processed data to and from the DSP functions.These objects are global in nature and are present throughout the time the simulation is being carried out. Each time a block is created on the screen (foreground), a corresponding object is created in the background. These objects are passed to the various DSP functions during the simulation of the processing operation.

The BLOCK objects have several fields for their operation.These include fields for storing output data which could be complex sequences, for storage of parameters for the function selected, linked lists for storage of addresses for input data to be used in processing functions and certain boolean variables for checking the various stages of operation.

When a BLOCK is created by the selection of a menu item , the corresponding dialog box for input of necessary parameters is called. These dialog boxes are based on the Window class TDialog (refer Appendix B of [4]) and their function is to obtain the various parameters necessary to simulate the block These parameters are then fed to the corresponding data objects in the background layer.

At the onset of a simulation run, the linked list of the background BLOCK objects is created reflecting the signal flow logic of the block diagram.The blocks are simulated in the order of their priority . This has been discussed in the next section . This simulation is done block by block with each block taking its input from the previous block connected to it..

2.3 Execution of the Simulator

At the outset user is presented with an opening menu from which he can start selecting the blocks he wants to use in the application . Each menu item has a help item which can be invoked from the context-sensitive help project to get information about the blocks the user wants to simulate. A speed-bar menu has been provided directly selecting some of the commonly occurring blocks .

When the user selects a particular block on the screen he will be asked to feed in the parameters required to simulate the running of the block. This is done in a dialog box specific to that block , asking for the parameters to be set. In case the user has any doubt regarding them then he can check the concerned help on that block from the dialog box itself using the mouse button . When the user feeds in the required parameters these are stored in the corresponding BLOCK objects of the background layer.

Now the user can move the blocks to convenient locations on the working window. There is a continous updation of the locations of these blocks in the corresponding objects of the foreground layer. This is necesary for the system to know the exact locations of the blocks on the working window for drawing subsequent connections between the blocks. He can connect the blocks in any order he desires. A point to note is that the user should first indicate the source block and then the destination block . The user is warned when a wrong connection is made. When a valid connection is made , the linked list of the foreground is updated with the appropriate block addresses. Visually a line with an arrow head is drawn on the screen indicating the connection made.

When the user has setup his block diagram and gives command to the system to 'run' the simulation.(refer Figure 2.1), the system first checks if all the connections are proper. If any missing connection is detected this will be indicated to the user. Once the system has finished its checking , the software will first ascertain from the user the number of samples upto which the simulation should occur. Then the objects of the background layer form their linked list as per the block diagram. The data-parameters in all the objects are initialized for the present simulation execution. The system keeps a record of all blocks connected by the userin the block diagram.The starting point for updation of the eventual run sequence is the first block in the list of connected blocks . Then a function 'RunBlocks' is called whose task is to identify the block to simulate next.

.. For understanding the working let us consider a block diagram which is shown in Figure 2.2. The numbers of the blocks show the chronological order of creation. Let us consider the first connection made was from block 4 to block 5. Hence the first block address given to 'RunBlocks' is of block 4. Hence the current block CB is block 4. Block 4 is a double input and double output block with addresses of block 1 in Previous_Block1or PB1, block 3 in Previous_Block 2 or PB2, block 5 in Next_Block1 or NB1and block 7 in Next_Block2 or NB2.

The flowchart of the function 'RunBlocks' is depicted in figure 2.3 . The first action in 'RunBlocks' is a sequence check mechanism to check whether any previous block or blocks exist to the current block CB and if so , have they been run or not . In the block diagram example shown in Figure3.2 the sequence check mechanism will check if block 1 (Previous_Block1) and then block 2 (Previous_Block1 of block 1) have been run.This will be run in the order block 2 followed by block 1. Next the sequence check will follow the path 4 - 3 (Previous_Block2) and run block 3. Then only block 4 will be executed. followed by block 5 (Next_Block.1) after its sequence check. Finally block 7 and block 6 will be executed after their respective sequence checks.Code segment of function 'RunBlocks' is given after the figures.

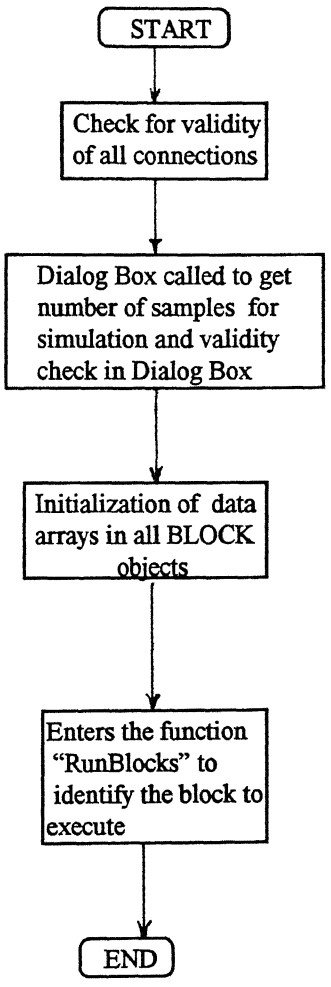
START
Check for validity of all connections
Dialog Box called to get number of samples for simulation and validity check in Dialog Box
Initialization of data arrays in all BLOCK objects
Enters the function "RunBlocks" to identify the block to execute
END

Figure 2.1 : Flow Chart for Execution of Run Sequence

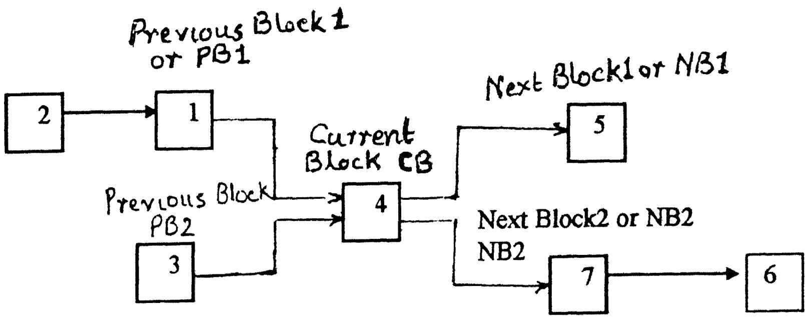

Figure 2.2 : Example of Block Diagram Connections

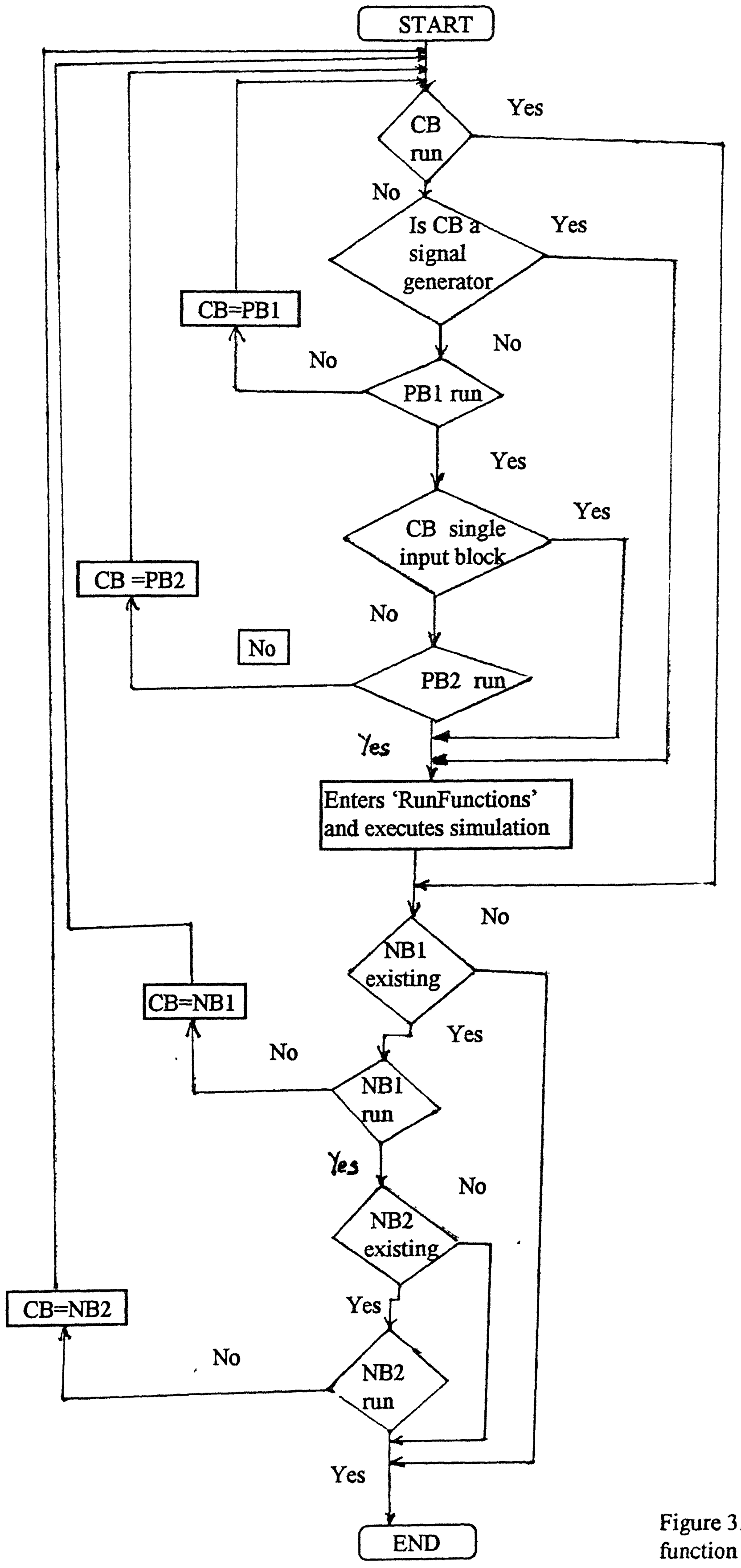

Figure 3.3:Flow Chart for function 'RunBlocks

```
void RunBlocks(BLOCK *BL,int Max) {

     if(BL->Ran!=TRUE)       / *  if block has not been simulated already  */
       if(BL->CodeNum<FIR)     /* if block is a signal generator block */
         {
           RunFunctions(BL,Max);
           if (BL->NextB1 != NULL)
             RunBlocks(BL->NextB1,Max);
          if (BL->NextB2 != NULL)
            RunBlocks(BL->NextB2,Max);
         }
     else
         {
          if ((BL->PrevB1)->Ran= =1)    /*  has previous block run?  */
                {
                 if((BL->PrevB2)= =NULL)  /* if it is a single input block  */
                   {
                      RunFunctions(BL,Max);
                      if (BL->NextB1 != NULL)          /* if a next connection exists  */
                         RunBlocks(BL->NextB1,Max);
                     if (BL->NextB2 != NULL)        /* if second next connection exists */
                         RunBlocks(BL->NextB2,Max);
                   }
        else    / ***    double input block   ***/
         {
           if ((BL->PrevB2)->Ran= =1)   /* Check if prev block2 has run or not */
               {
                 RunFunctions(BL,Max);
                if (BL->NextB1 != NULL)
                    RunBlocks(BL->NextB1,Max);
               if (BL->NextB2 != NULL)
                   RunBlocks(BL->NextB2,Max);
              }
        else
          RunBlocks(BL->PrevB2,Max);  /* run prev block2 first   */
       }
  }
 else
   RunBlocks(BL->PrevB1,Max);        /* run prev block1 first */
  }
 } // if BL->Ran !=1 case ends
}     /** END OF RUNBLOCKS  *****/
```

Once the simulation of the blocks has been done the user has the option to display the output of any block on a graph . when he chooses the block he wants to display, the corresponding BLOCK object is passed to the graphic module which gives the graphical display of the output . The user can save the output of any block in a file or if he wishes , he can change the parameters of any block he desires and do the simulation all over again . The user can run the simulation several times if he wishes and can add more blocks to the diagram at any stage he desires and do the simulation again . Description of the software has been given in the Users' Guide in Appendix A of this thesis and in Appendix B in [4] .

Having seen the execution of the simulation it is now appropriate to discuss the implementation of mathematical functions in the package . This is done in the next chapter .

CHAPTER 3

IMPLEMENTATION OF DSP AND MATHEMATICAL FUNCTIONS

A signal processing system can be split into certain fundamental groups of blocks each of which performs a particular mathematical operation.To simulate such a system there is a need to have certain input signal generators. Once the processing operation is completed there must be a module for display of outputs and for studying their characteristics. Thus the various blocks implemented in a simulation software can be broadly classified into the following categories.

- Signal generators
- DSP blocks
- Arithmetic blocks
- Transcendental blocks
- Windowing blocks
- Statistical Information

3.1 Signal Generators

These blocks are used to generate input data sequences for use during simulation. There are various kinds of input sequences which can be generated depending on the parameters set by the user for these blocks.Provision has also been provided for giving input through a file for any specific input sequence which the user wants to provide.These input sequences can be deterministic or random sequences.

3.1.1 Deterministic Signals.

- **Impulse.**

This sequence can be used to test the impulse response of any processing function block . It is determined by the relation

$$x(n) = A * \delta(n - m)$$

where A is a real constant and m is the delay in occurrence of the impulse.

- **Step Input.**

This sequence can be used to test the step response of any processing function block . It is determined by the relation

$$x(n) = A * u(n - m)$$

where A is a real constant and 'm' the is delay in start of the sequence.

- **Sinusoid.**

A sinusoid sequence is generated as per the equation

$$x(n) = A \sin\left(\frac{2\pi f n}{N} + \theta\right)$$

where A is the amplitude , N is the period in number of samples, and θ is the initial phase. This sequence is most useful in testing the frequency response of the functional blocks.

- **Damped Sinusoid.**

A damped sinusoid sequence is generated as per the equation

$$x(n) = A\, r^{n} \sin\left(\frac{2\pi f n}{N} + \theta\right)$$

where A is the amplitude , r is the decay factor and for r < 1 the sequence dies down as n increases ,N is the period in number of samples, and θ is the initial phase.

- **Ramp**.

This is the saw-tooth waveform which increases with .a definite slope and is periodic with a specific number of samples. This waveform is shown in the figure below

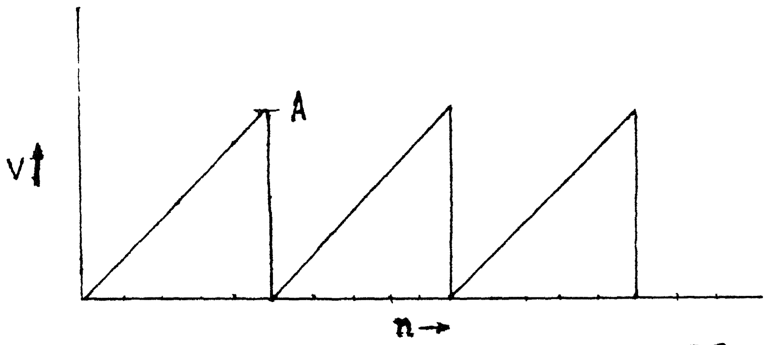

Figure 3·2

- **Pulse-Train**.

  This waveform is in the form of pulses with a definite period in number of samples and duty cycle as set by the user. A square wave can be generated by setting the duty cycle at 50% and the height of the pulses equal to the period in number of samples.

- **File and Keyboard Input**

The input can be provided by the user through file or keyboard which the system assumes as a complex input and can use this as another signal generator block.

3.1.2 Random Signals.

- There are three types of random sequences which can be generated. These are discussed below.

- **Uniform Random Sequence. [5]**

This sequence has its pdf given over the unit interval [0,1] which are unifomly distributed.The value of the pdf at any point is unity.

- **Gaussian Random Sequence**. [5]

This random sequence has its pdf given by gaussian distribution with mean 'm ' and variance σ^2 . This is generated by first considering two uniformly distributed random variables x_1 and x_2 on the interval [0,1] . Then using Box-Muller transformations we can get two normally distributed independent variables y_1 and y_2.

$$y_1 = \sqrt{-2 \ln x_1} \cos(2\pi x_2)$$

$$y_2 = \sqrt{-2 \ln x_1} \sin(2\pi x_2)$$

This can be converted to a gaussian random variable of mean m and variance σ^2 by the transformation

$$z = \sigma Y + m$$

where Y is the normally distributed random variable given by

$$Y = \sqrt{y_1^{\,2} + y_2^{\,2}}$$

- **Poisson Random sequence**.

The Poisson random variable gives the probability of acertain integer m of unit rate Poisson random events in a given time intervalof time X. This is related to the gamma distribution which gives the probability of waiting time between X and X+dX to the mth event in a Poisson random process of unit mean . The gamma distributed random variable can be generated by the expression

$$x = A \tan(\pi U) + B$$

where A and B are constants and U is a uniform random deviate between 0 and 1. In our simulation the value of A taken is = $\sqrt{2}$*mean and B = mean.

The Poisson random variable can be given by the probability of an integer j

$$\text{Prob}(j) = x^j e^{-x} / j!$$

By spreading this finite area in the spike at j over the range j to j+1 and and generating a non-integer deviate from the distribution , which is given by

$$q_x(m)dm = x\,[m]\, e^{-x} / [m]! \qquad \text{where } [m] \text{ is the largest integer} < m$$

would generate a Poisson random variable.

3.2 DSP Blocks

The signal processing blocks are the various stages in a DSP system in which the input is modified in order to enhance its qualities for better extraction of the information .These blocks form the core of the DSP simulation environment. They cover a multitude of operations which occur on the input data sequence . These processing blocks are discussed under further categorization of their functions.

- **FIR filter.**

The FIR filters are the most commonly used blocks in the DSP systems.They are specifically important in systems where linear phase characteristics have to be preserved.FIR filters can be realized in the time domain in either direct , cascade or lattice forms or in the frquency domain . However in this application only the direct form of FIR filter has been implemented. . Also , only time domain filtering is considered here . This is because when there is a simulation which involves feedback the application has to run the simulation sample by sample. Thus for such a process only a time domain implementation will be able to simulate this feature This is of the form

$$y(n) = \sum_{k=0}^{N-1} a(k)\, x(n-k)$$

where n = 0,1,.... and
N is the filter order.

where x(n) and y(n) are the input and output sequences . This system operates sample by sample.In modern day filtering applications large banks of FIR filters are implemented in tandem. This simulation package can also simulate these banks of filters operating in tandem.

- **IIR filters.**

The IIR filters are also widely used in DSP applications where phase characteristics need not be taken into consideration. Since these are recursive filters their output depends upon current and previous inputs and previous outputs , they can give a better performance as compared to nonrecursive filters with same number of coefficients. However the stability of IIR filters has to be ensured in its design itself by keeping all the poles in the transfer function within the unit circle itself. There are various forms for realizing the IIR filters. These are the direct form, cascade form or parallel form or in the pole-zero form. The implementation of IIR filter in the direct , cascade , parallel and pole - zero forms has been done in this application in the time domain .

In the **direct form** the IIR filter equation is implemented as

$$y(n) = \sum_{k=0}^{N-1} a(k) * x(n-k) - \sum_{m=1}^{N-1} b(m) * y(n-m) \qquad \text{where } n = 0,1,\ldots \text{ and } N \text{ is the filter order.}$$

x(n) and y(n) are the input and output sequences respectively.

In the **cascade form** the IIR filter transfer function is given as

$$H(z) = A * \prod_{k=1}^{N} \frac{1+ a(k,1)z^{-1} + a(k,2)z^{-2}}{1+b(k,1)z^{-1} + a(k,2)z^{-2}} \qquad \text{where N is the number of sections}$$

where the coefficients a(1,k) a(2,k) ,b(1,k) and b(2,k) are inputs for the each section and A is the gain constant.The sections are implemented in tandem in which each section is a second order block.

The **parallel form** of IIR filter implementation is given by the following expression

$$H(z) = \sum_{k=1}^{N} \frac{a(k,0) + a(k,1)z^{-1}}{1+b(k,1)z^{-1} + a(k,2)z^{-2}}$$ where N is the number of sections

This is realized as a summation of parallel connected blocks each of which has a second order transfer function.

In the **pole-zero realization** ,the IIR filter is specfied by its poles and zeroes.The expression for the filter transfer function is given below

$$H(z) = A \prod_{k=1}^{N} \frac{(1 - r(k)e^{j\theta(k)}z^{-1})(1 - r(k)e^{-j\theta(k)}z^{-1})}{(1 - R(k)e^{j\phi(k)}z^{-1})(1 - R(k)e^{-j\phi(k)}z^{-1})}$$ where N is the number of sections

This expression can be simplified further to give an expression to give

$$H(z) = A \prod_{k=1}^{N} \frac{(1 - 2\, r(k)\cos\theta(k)\, z^{-1} + r^2(k)z^{-2})}{(1 - 2\, R(k)\cos\phi(k)\, z^{-1} + R^2(k)z^{-2}}$$ where N is the number of sections

This expression has now been simplified into a cascade form implementation by considering the corresponding values for the coefficients of the cascade form as discussed above.

- **DFT and IDFT.**

There is always a need to view the spectral contents of a signal in order to understand its characteristics in larger measure . Hence there is a requirement to convert the signal into the frequency domain and vice versa . This is done by DFT and IDFT operations . The DFT of a signal is given by the equation

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \qquad \text{for } k = 0 \ldots N-1$$

where N is the length of the input sequence. The DFT maps N complex numbers into a new sequence of N complex numbers. This does not depend on any dimensionalquantity.This is a slow process as it requires a total of N^2 complex multiplications to be done. The inverse DFT is given by the following equation

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(n)ej2\pi\, kn/N \qquad \text{for } k = 0 \ldots N-1$$

The inverse DFT is also computed by the same algorithm after which the normalization is done.In the simulation , the same algorithm as given above has been implemented. .

- **FFT and IFFT.[6]**

This is a quicker method to convert a signal into the frequency domain as compared to the Discrete Fourier Transform and hence, has been aptly named as Fast Fourier Transform. There are various algorithms to compute the FFT of a sequence including FFT of radices 2 or higher powers of 2, by decimation in timeor frequency,FFTs of mixed radices not powers of 2. In this simulation package , FFT of radix 2 has been implemented which is based on decimation in time given by the Cooley-Tuke y algorithm.The advantage of FFT is that it reduces the number of computations to $(N/2)\log_2 N$ (to base 2) for an N point data

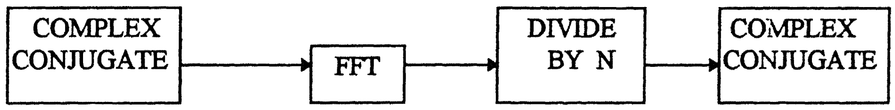

Figure 3 2 : Process of Computation of FFT

sequence.The inverse FFT is also computed using the FFT algorithm. The process of taking the IFFT of an input sequence is shown in the Figure 3.2 .

- **Convolution**

The convolution operation between two causal sequences is given by the equation

$$z(n) = \sum_{k=0}^{N-1} x(k)\ y(n-k) \qquad \text{where } n = 0\ldots N+L-1 \text{ , N,L are length of sequences and } k = 0 \ldots N-1 \text{ if } N < L$$

This is the convolution operation done in the time domain with real sequences.Convolution can also be achieved in the frequency domain.The product of the DFTs of two sequences if converted into time domain by inverse DFT will yield the same result.This can be shown as

$$z(n) = \text{IDFT}\ \{\ \text{DFT}\{x(n)\} * \text{DFT}\{y(n)\}\ \}$$

The simulation has both time and frequency domain convolution operations implemented for use by the user.

- **Correlation**

The correlation of two sequences x(n) and y(n) is given by the equation

$$z(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(k)\ y(n+k)$$

The software implements this operation. We can derive the autocorrelation of a sequence if the input sequence is given twice into this block during simulation.

- **Delay**

The delay block is utilized in DSP systems in order to delay the sequeces by specified number of samples. This block has widespread use in design of filters and in time domain convolution.This block is run sample by sample.

- **Median Filtering**

This block is used in some applications in order to smoothen the incoming data sequence. During simulation of this block , the median value occurring in the set of data , depending upon the window size of the median filter chosen is given as the output .

- **Interpolation and Decimation**

An analog signal is sampled at a fixed rate to obtain a discrete time sequence. If there emerges a need to change the sampling rate, then interpolation or decimation is required. Interpolation is used when there is a need to increase the sampling rate and decimation is used for decreasing the sampling rate. There are various methods of interpolation . In this simulation however, only linear interpolation has been implemented.

- **Gain :** $y(n) = K * x(n)$ where K is a real number

3.3 Arithmetic blocks.

These blocks perform certain important important functions in system design. The implementation of algebraic operations has been done for complex data sequences..All the operations occur in the blocks sample by sample.The following arithmetic operations have been implemented .

- **Addition** $z(n) = x(n) + y(n)$
- **Subtraction** $z(n) = x(n) - y(n)$
- **Multiplication** $z(n) = x(n) * y(n)$
- **Division** $z(n) = x(n) / y(n)$ if $y(n)$ is not 0.
- **Modulus** $y(n) = | x(n) |$
- **Power** $y(n) = [x(n)]^K$ where K is a real number. By choosing value of $K = 2$ a sequence can be squared or if $K = 0.5$ Then square root can be derived.
- **Conjugation** If $x(n) = x_1(n) + j[x_2(n)]$

  then $y(n) = x_1(n) - j[x_2(n)]$
- **Integration**

Integration is a sample by sample operation. There are numerous methods for numerical integration . In this simulation however, integration has been simulated using the Simpson'sRule.The samples are first taken to calculate the partial sums and then used to derive the final output for each sample.

3.4 Transcendental Blocks

All transcendental functions are designed to cater for complex data sequences except hyperbolic inverses . These functions have been implemented by the solution of a quadratic equation and operate only on real data .These blocks also operate sample by sample .

- **Sine** $y(n) = \sin(x(n))$
- **Cosine** $y(n) = \cos(x(n))$
- **Tangent** $y(n) = \tan(x(n))$
- **Sine Inverse** $y(n) = \arcsin(x(n))$
- **Cos Inverse** $y(n) = \arccos(x(n))$
- **Tan Inverse** $y(n) = \arctan(x(n))$
- **Sin Hyperbolic** $y(n) = \sinh(x(n))$
- **Cos Hyperbolic** $y(n) = \cosh(x(n))$
- **Tan Hyperbolic** $y(n) = \tanh(x(n))$
- **Sinh Inverse** $y(n) = \log_e(x(n) + \sqrt{[x(n)]^2 +1})$
- **Cosh Inverse** $y(n) = \log_e(x(n) \}+ \sqrt{[x(n)]^2 -1})$
- **Tanh Inverse** $y(n) = 0.5 * \log_e[(1 + x(n)) / (1 - x(n))]$
- **Logarithm** $y(n) = \log_K(x(n))$ where k is a real number.
- **Exponential** $y(n) = \exp(x(n))$

3.5 Windowing.

Sequences tend to be very large or can be even infinite in DSP.However only a part of the waveform may be required for analysis These techniques are achieved by windowing techniques.. The commonly used window functions have been implemented in our simulation. The following windowing functions have been implemented.

- **Rectangular Window**

$$w(n) = \begin{cases} = 1 & , \quad 0 <= n <= N-1 \\ = 0 & , \quad \text{otherwise} \end{cases}$$

- **Bartlett Window.**

$$w(n) = \begin{cases} = \frac{2n}{N-1} & , \quad 0 <= n <= \frac{N-1}{2} \\ = 2 - \frac{2n}{2} & , \quad \frac{N-1}{2} <= n <= N \\ = 0 & , \quad \text{otherwise} \end{cases}$$

- **Blackman Window**

$$w(n) = \begin{cases} 0.42 - 0.5 \cos \frac{2\pi n}{N-1} + 0.08 \cos \frac{4\pi n}{N-1} & , \quad 0<=n<= N-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

- **Hanning Window**

$$w(n) = \begin{cases} 0.5 \{ 1 - \cos \frac{2\pi n}{N-1} \} & , \quad 0<=n<= N-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

- **Hamming Window**

$$w(n) = \begin{cases} 0.54 - 0.46 \cos \frac{2\pi n}{N-1} \} & , \quad 0<=n<= N-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

- **Kaiser Window**

$$w(n) = \begin{cases} \dfrac{I_0\left\{ w_a \sqrt{\left[\dfrac{N-1}{2}\right] - \left[n - \dfrac{N-1}{2}\right]^2} \right.}{I_0\left[w_a \left(\dfrac{N-1}{2} \right]\right.} & , \quad 0<=n<= N-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

where I_0 ( ) is the modified zeroth order Bessel function of the first kind and typical values of $w_a((N-1)/2)$ are in a range $4< w_a((N-1)/2) < 9$.

3.6 Statistical Information

Statistical information is displayed by selection of another memu item. This is given for generating statistics for any block which has been simulated. The following information can be generated .

Second Order Statistics

The following information is generated when this option is selected.

- Mean
- Second moment
- Variance
- Standard deviation
- RMS Value
- Minimum value of sequence
- Maximum value of sequence
- Absolute minimum of sequence
- Absolute maximum value of sequence

Nth order statistics

The following information is displayed under this option.

- Nth moment
- Nth central moment

The system can be run several times to simulate the blocks upto a maximum sample value of 1024 . We now consider an example of the simulation and see the results of the simulation run . Refer to the Users'Guide in Appendix A and Appendix B of [4] for further details in order to carry out simulation.

3.7 Simulation Example

We consider a simple example in which two sinusoids of frequencies 0.09 and 0.35 of nearly equal amplitude are added . The signal is now given to a low pass FIR filter of order 23 whose cut off frequency is 0.23 . The filtered output is given to an FFT block . The designed block diagram is shown in Figure 3.3 . The magnitude of the FFT of the input to FIR filter is shown in Figure 3.4 When the magnitude of the output of the FFT is taken and the display plotted we clearly see that the value of the output at frequency 0.35 has been scaled down considerably thereby showing the low. pass filter action of filtering out the unwanted frequency.This is shown in the Figure 3.5 .Thus the software can be used for simulations and these can be viewed on the graphical display.

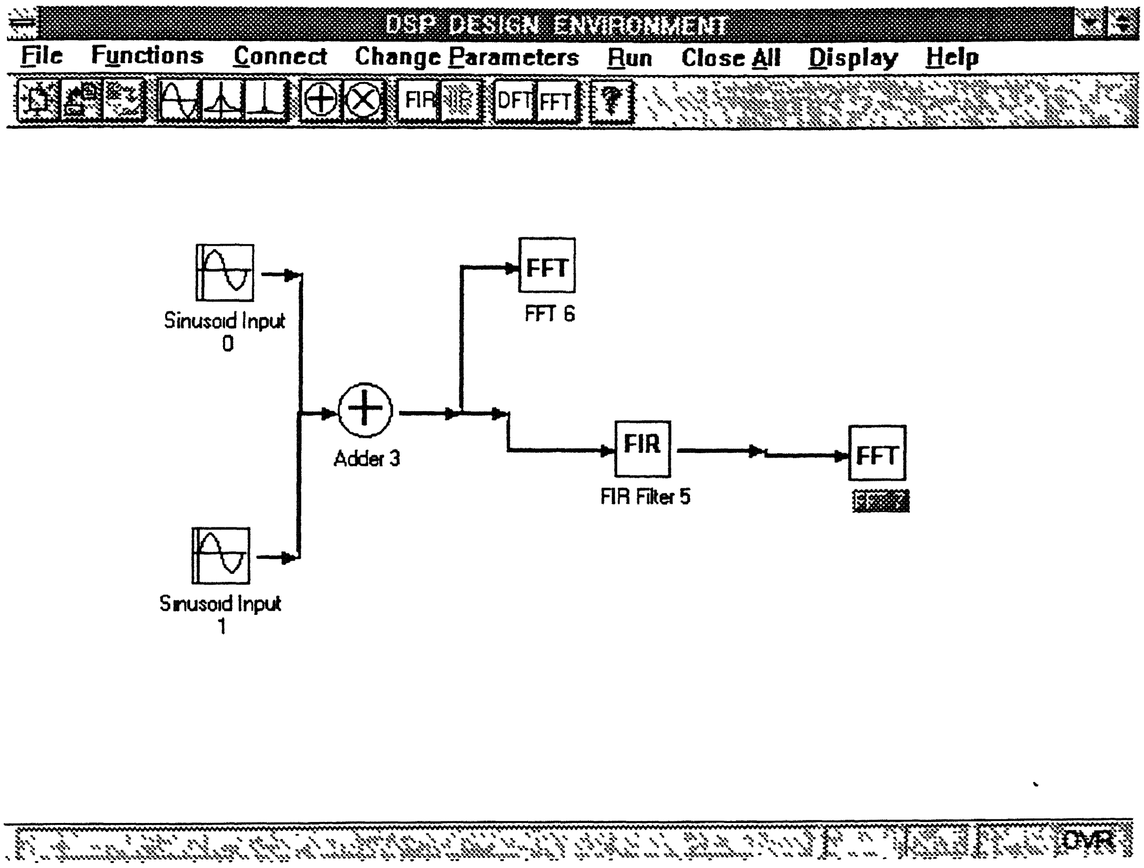

Figure 3.3 : Block Diagram for Simulation

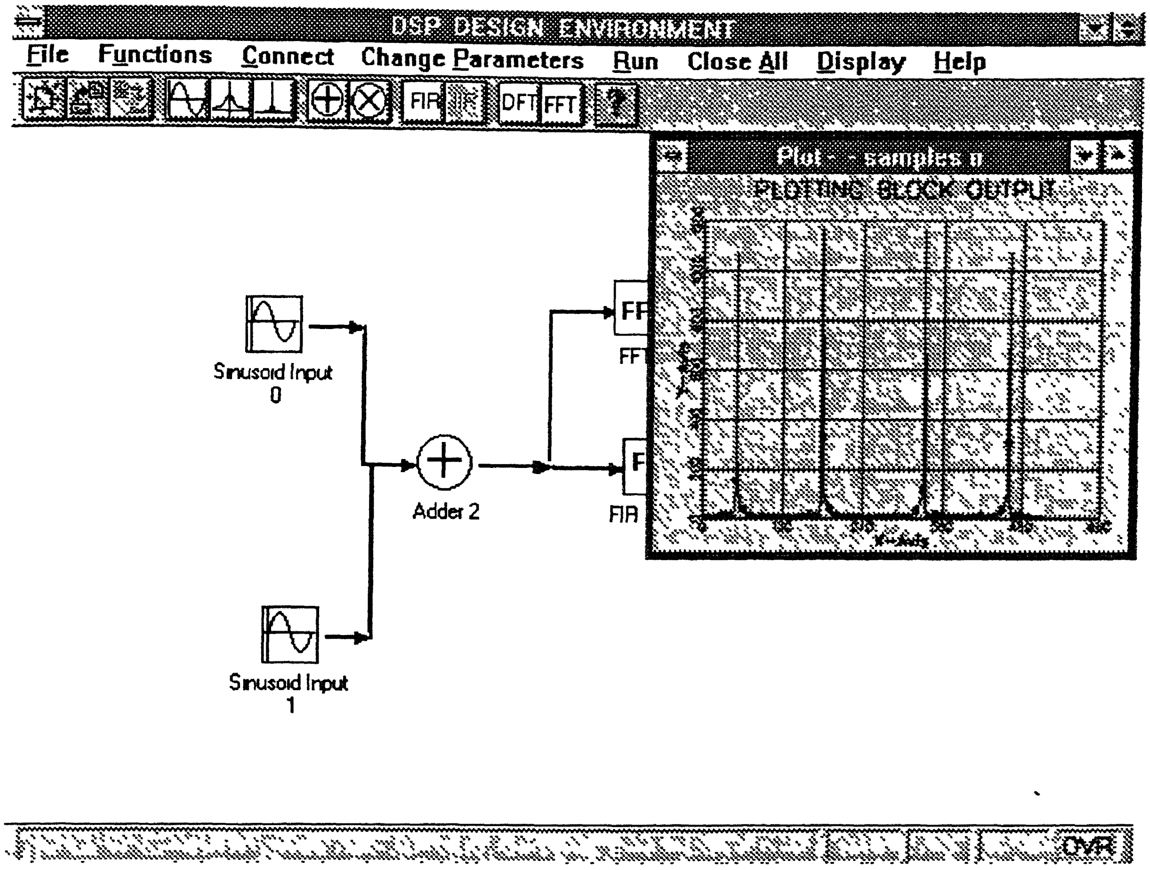

Figure 3.4 : FFT of input to FIR Filter Block

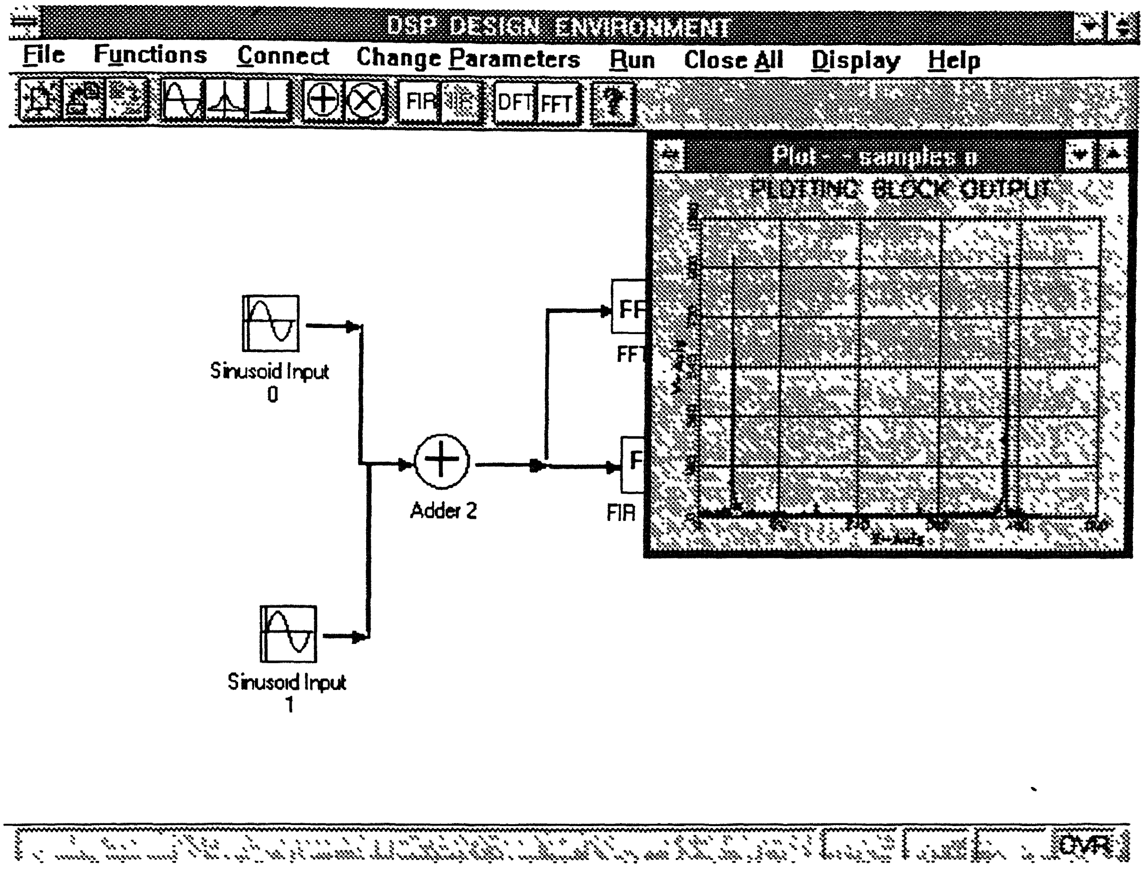

Figure 3.5 : FFT of output of FIR Filter Block

CHAPTER 4

ENHANCEMENTS IN THE SOFTWARE

The software developed has a large scope for improvement in the several respects . A large number of additions can still be made in order to enhance the mathematical functions and system capabilities of the application . Some of the enhancements which are concerned with the computational aspects are discussed below .

4.1 Enhancing the function library

Presently only a certain number of basic DSP and mathematical functions have been added . There is considerable scope for increasing the function library. Some of the types of functions which can be added without modifying the data structures are given below.

- Processing functions like clipping , thresholding and accumulation of data which can form a separate block by themselves .
- More number of arithmetic functions can be added for handling of real and complex data like conversion of real to complex data and vice versa..
- Larger number of signal generator functions can be added like triangular wave , sinc and gaussian pulse inputs to name a few types of generators .
- Functions needed in image processing can also be incorporated into the application which will make it more useful .

It would be in order to mention the steps to be taken for implementing a new function . The function code would have to be in C language code and capable of handling the data structure . Let us consider an example for this and see how to implement this in the program .Let us consider that we want to add a new block 'Clip' in the software .

- The function block must be allotted a new code number in the include file codenum h..

- Each new function block has to be called from the menu . This can be done by adding a new menu item 'Clip' in the Popup menu under 'DSP Functions' . This will be done in the resource file 'b3app rc' by invoking the Resource Workshop [4] .

- A new icon has to be created and stored in the working directory . This can be created as a file 'clip.ico' using the Resource Workshop and declared as a constant in the resource file . For details regarding this please refer [4].

- There is a requirement of taking the upper and lower clipping values from the user. Hence a dialog box will have to be created using the Resource Workshop in the resource file 'b3app.rc' for input of these two values . The dialog box name has to be defined as a constant in the resource header file 'b3app.rh'.

  The function called ,say 'CmClip' ,when the menu item 'Clip' is selected and should be a member function of the class 'b3MDIClient' [4] has to be written .This function should contain a call to the function 'Create_Child_block' in order to create the child window and paste its icon on it and function 'NewDataBlock'which creates a data object and initializes it.. Then the function 'CallDialog' should be called to create the dialog box when this menu item is selected.

- The function 'CallDialog' should have an addition of one more case statement for 'Clip in which the corresponding dialog box is called . The inputs given are read by the function 'GetDlgInput' in which another case statement would be added for reading the data input from the dialog box to the appropriate fields of the data structure.

- The function is executed when its turn comes during the simulation . The call to the function is done through a function 'RunFunctions' in which another case statement would be added for thecall to the function 'clip'.

- Another case statement would be added in the function 'graf_names' for adding the strings for the heading , title , x-axis and y axis for depicting on the graphic display of the output .

- These are the basic steps to be followed when a new function is to be added in the software.

4.2 System enhancements

- The system is at present working for a maximum of 1024 samples . Every block requires to store a maximum number of that many samples and hence the requirement of memory to be allocated for this in each simulation block becomes a rigid condition even if it may not be used at all. The number of samples can be be increased to higher values but presently the limitation of available memory is causing the program to abort when the number of simulation blocks increases and we repeatedly carry out the simulation over several runs . However , this problem is being tackled by using global allocation of memory for the data and the functions by declaring them as 'far'.
- Creation of a file system is a required enhancement where the user can include already designed set of blocks as a single block in more complex DSP systems . This can be done by first making the program able to store a system on the hard disk . This requires the diagram , the data blocks and their respective connections to be stored in separate files. These need to be read together and handled appropriately only when they are invoked by a user . For more details refer [4].
- Another feature which needs to be done is to have the facility of multiple runs. The user should specify the number of runs which the system should do . This will increase the utility of the software for statistical information and for including aspects of spectral estimation
- Code generation for DSPs can be advanced feature which can be built into the software. When a block diagram is created by a user , the software provides the code for simulation of the blocks in the assembly language of a processor like the ADSP-2100 . These can be downloaded to a processor by the host and the simulation executed on this processor . This could also have the facility of making the library routines visible in both

code and by use of suitable break points to show the functioning of the processor also , by display of the register contents in suitable display windows made for the purpose .

CHAPTER 5

CONCLUSION

The simulation system developed is capable of simulation of a large number of DSP applications . It is capable of multiple number of runs and it possesses a good graphical display module . It also has a set of statistical information generation routines which are used in post processing for analysis of output of various blocks . Thus it can be used for block diagram simulation of DSP systems .

The software is an extremely user friendly one and a user can easily learn to operate it by referring to the Users' Guide . In addition , Help is provided in the software itself . There is a string table displayed under the main window which tells the nature of the function . In addition there are Help files in the project which are accessed when the user presses F1 button . The dialog boxes also have a corresponding 'help ' provided which are accessed when the Help button is pressed .

Once the enhancements mentioned in the previous chapter are incorporated into the software it will become extremely useful for design purposes . It will be possible to test reasonably complex simulation problems as it a fully dedicated softwareand observe the results . It can be used as a software tool to perform experiments and present results of the simulation performed . It can be an inexpensive way to achieve block diagram simulation of DSP systems.

Bibliography

[1] Kornbluh, Ken , "Active Data Analysis: Advanced software for the 90s",IEEE Spectrum, Nov 94.

[2] Lee E.A, Ho Wai-Hung, Goei E.E, Bier J.C, Bhattacharyya S, " Gabriel : A Design Environment for DSP", IEEE Transactions on Acoustics, Speech, and Signal Processing Vol 37, No 11, Nov 89.

[3] Braham R , "Application Software", IEEE Spectrum Jan 95.

[4] Sharma , Major VP,"Windows Based Block Diagram Simulation : Implementation of Object Windows Interface", M Tech Thesis , IIT Kanpur, April 95.

[5] Press WH, Flannery BP, Teulolsky SA, Vetterling WT, "Numerical Recipes in C : The Art of Scientific Computing", Cambridge University Press , !988.

[6] Embree PM, Kimble Bruce, "C Language Algorithms for Digital Signal Processing " Prentice Hall Publications , 1991.

[7] Gallagher R.S, "Visualization : The look of reality" , IEEE Spectrum Nov 94

[8] Shanmugham, K Sam, "Simulation and Implementation of Communication and Signal Processing Systems", IEEE Communications Magazine, July 94.

.[9] Oppenheim A, Schafer R,"Discrete Time Signal Processing ", Prentice Hall India 1993.

[10] Proakis J, Manolakis D , “Introduction to Signal Processing”, MacmillanPublishing Company , 1988.

[11] Conger James, “ Windows API Bible”, Waite Group , 1993.

[12] Petzold , Charles , “Programming Windows 3.1”, Microsoft Press, 1992.

[13] Conger James, “Windows Primer Plus“, Waite Group, 1993.

[14] Prata ,Stephen “C++ Primer Plus”, Waite Group , 1992.

[15] Borland C++ Programmers’ Guide, Borland International Inc., 1993.

[16] Borland C++ Object Windows Reference Manual , Borland International Inc, 1993.

Appendix A

Users' Guide for DSP and Mathematical Functions

All the DSP and mathematical blocks are listed in the Popup menu which appears when the menu item Functions is clicked . The various blocks implemented in a simulation software can be broadly classified into the following categories each of which forms a sub-menu itself.

- Signal generators
- DSP blocks
- Arithmetic blocks
- Transcendental blocks
- Windowing blocks
- Statistical blocks

A1 Signal Generators

The following blocks are used to generate input data sequences for use during simulation.

- **Impulse.** This sequence is determined by the relation

  $$x(n) = A * \delta(n - m)$$

  where A is a real constant and m is the delay in occurrence of the impulse The input of parameters is done through a dialog box which appears when this block is selected from the system menu or if 'Change Parameters' command is chosen for this block . The following inputs are taken from the user.

- Height of impulse A, a real value.
- 'm' , the sample number at which impulse occurs.

- **Step Input.** This sequence is determined by the relation

$$x(n) = A * u(n-m)$$

where A is a real constant and 'm' the is delay in start of the sequence. The input of parameters is done through a dialog box which appears when this block is selected from the system menu or if 'Change Parameters' command is chosen for this block .

- Height of impulse A, a real value.
- 'm' , the sample number at which step begins.

- **Sinusoid.** A sinusoid sequence is generated as per the equation

$$x(n) = A \sin\left(\frac{2\pi f n}{N} + \theta\right)$$

where A is the amplitude , N is the period in number of samples, and θ is the initial phase. When this block is chosen from the menu a dialog box appears in which the following data is to be given

- Amplitude A should be a real number.
- frequency f , such that $0.0 < f < 0.5$.
- and the initial phase θ in degrees .

- **Damped Sinusoid.** A damped sinusoid sequence is generated as per the equation

$$x(n) = A\ r^n \sin\left(\frac{2\pi f n}{N} + \theta\right)$$

where A is the amplitude , r is the decay factor and for $r < 1$ the sequence dies down as n increases ,N is the period in number of samples, and θ is the initial phase in degrees When this block is chosen from the menu a dialog box appears in which the following data should be given.

- Amplitude A a real number.
- Decay factor $r < 1.0$
- Frequency f, such that $0.0 < f < 0.5$
- Initial phase θ in degrees

- **Ramp**. This is the saw-tooth waveform which increases with .a definite slope upto

maximum value A and is periodic with a specific number of samples N . When this block is selected a dialog box appears in which the above parameters are asked.

- Maximum Value attained A , a real value.
- Period of sawtooth N in number of samples.

- **Pulse-Train**. This waveform is in the form of pulses with a definite period and duty cycle as set by the user. When this block is selected a dialog box appears in which the the following parameters are asked.
  - height of the pulse A , a real value
  - Period of the pulse N in number of samples
  - Duty cycle in percentage .

- **File Input**

The input can be provided by the user through a file . The user has to give the filename and select the path in the dialog box appearing on selection of this block.

- **Keyboard Input**

The input can be given through the keyboard . The number of data points has to be specified first in a dialog box. Then the system will accept input of that many complex or real data samples.

- **Uniform Random Sequence**. This block generates a random sequence uniformly distributed over the unit interval [0,1]. The input required for this block in the dialog box is:
  - Seed integer which should be **positive.**
- **Gaussian Random Sequence**. This block generates a sequence of real values which have a gaussian distribution. The inputs required are taken in the dialog box appearing on selection of this block.
  - Mean
  - Variance
  - Seed integer which should be positive. This seed integer value can be changed when more than one execution of the simulation is done by choosing the 'Change Parameters' menu
- **Poisson Random Sequence**. This block generates a sequence of real values which have a Poisson distribution. The inputs required for this block are
  - Mean

- Seed integer which should be positive. This seed integer value can be changed when more than one execution of the simulation is done by choosing the 'Change Parameters' menu option.

A2 DSP Functions

The following DSP function blocks have been implemented .

- **FIR filter**. The direct form of FIR filter has been implemented in time domain . This is of the form

$$y(n) = \sum_{k=0}^{N-1} a(k)\ x(n-k) \qquad \text{where } n = 0....N-1 \quad \text{and}$$
$$N \text{ is the filter order.}$$

where x(n) and y(n) are the input and output sequences . The input of coefficients can be done by file or from keyboard in a dialog box. The sequence to be followed is given below

- First a dialog box appears in which the user has to feed the filter order .
- Next the user is asked an option for file input or input through keyboard .
- If he chooses the File option then he has to give the filename and the correct path for it in the dialog box which appears.The system will read the file and get the necessary coefficients . The data in file should be in ASCII form and should contain no special characters.
- If he chooses the Keyboard option then a dialog box will appear to take the coefficients in batches of five each time .

- **IIR Filter (Direct Form)**

In the direct form the IIR filter equation is given as

$$y(n) = \sum_{k=0}^{N-1} a(k) * x(n-k) - \sum_{m=1}^{N-1} b(m) * y(n-m) \quad \text{where } n = 0....N-1 \quad \text{and}$$

N is the filter order.

x(n) and y(n) are the input and output sequences respectively . The sequence to be followed is given below.

- At first a dialog box appears in which the user has to feed the filter order .
- Next the user is asked an option for file input or input through keyboard .
- If he chooses the File option then he has to give the filename and the correct path for it in the dialog box which appears.The system will read the file and get the necessary coefficients . The data in file should be in ASCII form and should contain no special characters.
- If he chooses the Keyboard option then a dialog box will appear to take the coefficients in batches of five each time first for the numerator coefficients and then for the denominator coefficients..

- **IIR Filter (Cascade Form)**

In the cascade form the IIR filter transfer function is given as

$$H(z) = A \prod_{k=1}^{N} \frac{1+ a(k,1)z^{-1} + a(k,2)z^{-2}}{1+b(k,1)z^{-1} + a(k,2)z^{-2}} \quad \text{where N is the number of sections}$$

where the coefficients a(1,k) a(2,k) ,b(1,k) and b(2,k) are inputs for the each section.The sections are implemented in tandem in which each section is a second order block.The sequence followed is given below.

- First a dialog box appears in which the user has to feed the number of sections .

- Then the user is asked an option for file input or input through keyboard
- If the user chooses the File option then he has to give the filename and the correct path for it in the dialog box which appears.The system will read the file and get the necessary coefficients . The data in file should be in ASCII form and should contain no special characters.
- If he chooses the Keyboard option then a dialog box will appear to take the coefficients section by section .
- Another dialog box will appear at the end of these to take the value of A, the normalization constant.

- **IIR Filter (Parallel Form)**

  The parallel form of IIR filter implementation is given by the following expression

$$H(z) = \sum_{k=1}^{N} \frac{a(k,0) + a(k,1)z^{-1}}{1+b(k,1)z^{-1} + a(k,2)z^{-2}} \qquad \text{where N is the number of sections}$$

This is realized as a summation of parallel connected blocks each of which has a second order transfer function. The sequence to be followed is given below.

- First a dialog box appears in which the user has to feed the number of sections .
- Then the user is asked an option for file input or input through keyboard .
- If the user chooses the File option then he has to give the filename and the correct path for it in the dialog box which appears.The system will read the file and get the necessary coefficients . The data in file should be in ASCII form and should contain no special characters.
- If he chooses the Keyboard option then a dialog box will appear to take the coefficients section by section .

- **IIR Filter (Pole-Zero Form)**

In the pole-zero realization ,the IIR filter is specfied by its poles and zeroes.The expression for the filter transfer function is given below

$$H(z) = A \prod_{k=1}^{N} \frac{(1 - r(k)e^{j\theta(k)}z^{-1})(1 - r(k)e^{-j\theta(k)}z^{-1})}{(1 - R(k)e^{j\phi(k)}z^{-1})(1 - R(k)e^{-j\phi(k)}z^{-1})}$$

where N is the number of sections

The sequence to be followed is given below.

- First a dialog box appears in which the user has to feed the number of sections .
- Then the user is asked an option for file input or input through keyboard .
- If the user chooses the File option then he has to give the filename and the correct path for it in the dialog box which appears.The system will read the file and get the necessary coefficients . The data in file should be in ASCII form and should contain no special characters.
- If he chooses the Keyboard option then a dialog box will appear to take the coefficients section by section .
- Another dialog box will appear at the end of these to take the value of A, the normalization constant.

- **DFT**

The DFT of a signal is given by the equation

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \qquad \text{for } k = 0\ldots N-1$$

where N is the length of the input sequence. In this block the DFT is taken to the length of the number of samples for which the simulation is being done.

- **IDFT**

The inverse DFT is given by the following equation

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)e^{-j2\pi\, kn/N} \qquad \text{for } k = 0 \ldots N-1$$

The IDFT is also computed by the same algorithm as DFT after which the normalization is done .

- **FFT**

In this simulation package , FFT of radix 2 has been implemented which is based on decimation in time given by the Cooley-Tukey algorithm.

- **IFFT**

The inverse FFT is computed using the FFT algorithm only which is FFT of radix 2 based on decimation in time given by the Cooley-Tukey algorithm.

- **Convolution (Time Domain)**

The convolution operation between two causal sequences is given by the equation

$$z(n) = \sum_{k=0}^{N-1} x(k)\ y(n-k) \qquad \text{where } n = 0 \ldots N-1$$

This is the convolution operation done in the time domain with real sequences. There is a requirement of two input connections for this block to correctly operate.

- **Convolution (Frequency Domain)**

Convolution in the frequency domain has been implemented as

$$z(n) = \text{IDFT}\ \{\ \text{DFT}\{x(n)\} * \text{DFT}\{y(n)\}\ \}$$

- **Correlation**

The correlation of two sequences x(n) and y(n) is given by the equation

$$z(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(k)\ y(n+k)$$

The software implements this operation. We can derive the autocorrelation of a sequence if the same block is connected twice into this block during simulation.

- **Delay**

This block when selected will delay the input sequence by a specified number of samples. When this block is selected a dialog box appears to feed the parameter into the system.

- Number of samples to be delayed.This can be changed by invoking the 'Change Parameters' option.

- **Median Filtering**

In this block during simulation , the median value occurring in the set of data , depending upon the window size of the median filter chosen is given as the output . When this block is selected a dialog box appears in which the input is to be given.

- Filtering window size has to be given by the user as an integer value. He can change the window size by using the 'Change Parameters' menu option.

- **Interpolation**

This block performs linear interpolation on the input sequence . When this block is selected a dialog box appears in which the input has to be given by the user

- Interpolation factor has to be given by user as an integer value.He can change this value by using the 'Change Parameters' menu option.

- **Decimation**

This block performs decimation on the input sequence . When this block is selected a dialog box appears in which the input has to be given by the user

- Decimation factor has to be given by user as an integer value He can change this value by using the 'Change Parameters' menu option.

- **Gain :** $\{ y(n) \} = K * \{ x(n) \}$ where K is a real number

A3 Arithmetic blocks.

.The following arithmetic operations have been implemented for complex input sequences.

- **Adder** $\{ z(n) \} = \{ x(n) \} + \{ y(n) \}$
- **Subtractor** $\{ z(n) \} = \{ x(n) \} - \{ y(n) \}$
- **Multiplier** $\{ z(n) \} = \{ x(n) \} * \{ y(n) \}$
- **Divider** $\{ z(n) \} = \{ x(n) \} / \{ y(n) \}$
- **Modulus** $\{ y(n) \} = | \{ x(n) \} |$
- **Power** $\{ y(n) \} = \{ x(n) \}^K$ where K is a real number.This value is taken in a dialog box when this block is selected. This value can be changed when the 'Change Parameters' option ischosen by the user.
- **Conjugation** If $\{ x(n) \} = \{ x_1(n) \} + j\{ x_2(n) \}$

  then $\{ y(n) \} = \{ x_1(n) \} - j\{ x_2y(n) \}$
- **Integration** Integration has been implemented using the Simpson's Rule.

A4 Transcendental Functions

All transcendental functions are designed to cater for complex data sequences except hyperbolic inverses.

- **Sine** $\{ y(n) \} = \sin(\{ x(n) \})$
- **Cosine** $\{ y(n) \} = \cos(\{ x(n) \})$
- **Tangent** $\{ y(n) \} = \tan(\{ x(n) \})$

- **Sine Inverse** $\{ y(n) \} = \arcsin(\{ x(n) \})$
- **Cos Inverse** $\{ y(n) \} = \arccos(\{ x(n) \})$
- **Tan Inverse** $\{ y(n) \} = \arctan(\{ x(n) \})$
- **Sin Hyperbolic** $\{ y(n) \} = \sinh(\{ x(n) \})$
- **Cos Hyperbolic** $\{ y(n) \} = \cosh(\{ x(n) \})$
- **Tan Hyperbolic** $\{ y(n) \} = \tanh(\{ x(n) \})$

- **Sinh Inverse** $\{ y(n) \} = \log_e(\{ x(n) \} + \sqrt{ \{ x(n)\}^2 +1 })$
- **Cosh Inverse** $\{ y(n) \} = \log_e(\{ x(n) \} + \sqrt{ \{ x(n)\}^2 -1 })$
- **Tanh Inverse** $\{ y(n) \} = 0.5 * \log_e[1 + \{ x(n) \}) / (1 - \{x(n) \})]$
- **Logarithm** $\{ y(n) \} = \log_K(\{ x(n) \})$ where k is a real number.The value of k is taken by a dialog box which appears when the block is selected.This base value can be changed by using the 'Change Parameters' option.
- **Exponential** $\{ y(n) \} = \exp(\{ x(n) \})$

A5 Windowing Blocks.

. The following windowing functions have been implemented. When any window function is selected then a dialog box appears which takes input for the starting sample number and the length of the window. These values can be changed by choosing the 'Change Parameters' option.

- **Rectangular Window**

$$w(n) = \begin{cases} = 1 & , \quad 0 <= n <= N-1 \\ = 0 & , \quad \text{otherwise} \end{cases}$$

- **Bartlett Window.**

$$w(n) = \begin{cases} = \dfrac{2n}{N-1} & , \quad 0 <= n <= \dfrac{N-1}{2} \\ = 2 - \dfrac{2n}{2} & , \quad \dfrac{N-1}{2} <= n <= N \\ = 0 & , \quad \text{otherwise} \end{cases}$$

- **Blackman Window**

$$w(n) = \begin{cases} 0.42 - 0.5 \cos \dfrac{2\pi n}{N-1} + 0.08 \cos \dfrac{4\pi n}{N-1} & , \quad 0<=n<= N-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

- **Hanning Window**

$$w(n) = \begin{cases} 0.5 \left\{ 1 - \cos \dfrac{2\pi n}{N-1} \right\} & , \quad 0<=n<= N-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

- **Hamming Window**

$$w(n) = \begin{cases} 0.54 - 0.46 \cos \dfrac{2\pi n}{N-1} \} & , \quad 0<=n<= N-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

- **Kaiser Window**

$$w(n) = \begin{cases} \dfrac{I_0\{ w_a \sqrt{\left[\dfrac{N-1}{2} \right] - \left[n - \dfrac{N-1}{2} \right]^2}}{I_0[w_a (\dfrac{N-1}{2}]} & , \quad 0<=n<= N-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

where I_0 ( ) is the modified zeroth order Bessel function of the first kind and typical values of $w_a ((N-1) / 2)$ are in a range $4 < w_a ((N-1)/2) < 9$.

A6 Statistical Information

A separate menu item has statistical information being displayed. This is given for generating statistical information . The following information can be generated .

Second Order Statistics

The following information is generated when this option is selected.

- Mean

$$X = \frac{1}{N} \sum x_i \qquad \text{where } I = 0,1, \ldots , N-1$$

- Second moment

$$Y = \frac{1}{N} \sum x_i^2 \qquad \text{where } I = 0,1, \ldots , N-1$$

- Variance $\sigma^2 = Y - X^2$
- Standard deviation $= \sigma$
- RMS Value $= \sqrt{Y}$
- Minimum value of sequence
- Maximum value of sequence
- Absolute minimum of sequence
- Absolute maximum value of sequence

Nth order statistics

A dialog box appears for taking the order of statistics required which is greater than 2.

The following information is displayed.

- Nth moment

$$Z = \frac{1}{N} \sum X_i^{\,n} \qquad \text{where I} = 0, 1, \ldots, N-1$$

- Nth central moment $= Z - X^n$

NOTE : For details regarding the operation of the software refer Appendix B of [4].